

IT Honours Research Project Coversheet

To be attached to the front of the assessment.

|  |  |
| --- | --- |
| **Campus:** | **\_Midrand\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Faculty:** | **\_IT\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Module Code:** | **\_ITDAA4-12\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Lecturer’s Name:** | **\_Mokgapi London Mashabela\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**Student Full Name: \_Melany Opperman\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Student Number: \_6P6NPJX46\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |
| --- | --- | --- |
| **Indicate** | **Yes** | **No** |
| Plagiarism report attached | x |  |

**Declaration:**

I declare that this assessment is my own original work except for source material explicitly acknowledged. I also declare that this assessment or any other of my original work related to it has not been previously, or is not being simultaneously, submitted for this or any other course. I am aware of the AI policy and acknowledge that I have not used any AI technology to generate or manipulate data, other than as permitted by the assessment instructions. I also declare that I am aware of the Institution’s policy and regulations on honesty in academic work as set out in the Conditions of Enrolment, and of the disciplinary guidelines applicable to breaches of such policy and regulations.

|  |  |
| --- | --- |
| **Signature**  **M.Opperman** | **Date**  **21/05/2024** |

Eduvos (Pty) Ltd. (formerly Pearson Institute of Higher Education) is registered with the Department of Higher Education and institution under the Higher Education Act, 101, of 1997. Registration Certificate number: 2001/HE07/008Training as a private higher education

**Lecturer’s Comments:**

|  |  |  |
| --- | --- | --- |
| **Marks Awarded:** |  | **%** |
|  |  |  |
| **Signature** | **Date** | |

Table of Contents

[Section A 5](#_Toc168754583)

[Question 1 5](#_Toc168754584)

[a. Code Snippet 5](#_Toc168754585)

[b. Code Output 6](#_Toc168754586)

[c. Output explanation 6](#_Toc168754587)

[Question 2 7](#_Toc168754588)

[2.1.a. 7](#_Toc168754589)

[a. Code Snippet 7](#_Toc168754590)

[b. Code Output 7](#_Toc168754591)

[c. Output explanation 8](#_Toc168754592)

[2.1.b. 9](#_Toc168754593)

[a. Code Snippet 9](#_Toc168754594)

[b. Code Output 9](#_Toc168754595)

[c. Output explanation 10](#_Toc168754596)

[2.1.c. 11](#_Toc168754597)

[a. Code Snippet 11](#_Toc168754598)

[b. Code Output 11](#_Toc168754599)

[c. Output explanation 12](#_Toc168754600)

[Question 3 13](#_Toc168754601)

[3.1. 13](#_Toc168754602)

[a. Code Snippet 13](#_Toc168754603)

[b. Code Output 14](#_Toc168754604)

[c. Output explanation 14](#_Toc168754605)

[3.2. 16](#_Toc168754606)

[a. Code Snippet 18](#_Toc168754607)

[b. Code Output 19](#_Toc168754608)

[c. Output explanation 19](#_Toc168754609)

[Question 4 21](#_Toc168754610)

[a. Code Snippet 21](#_Toc168754611)

[b. Code Output 23](#_Toc168754612)

[c. Output explanation 24](#_Toc168754613)

[References 25](#_Toc168754614)

## Section A

## Question 1

### Code Snippet

1. #ITDAA4-12 QUESTION 1
2. #Importing the libraries needed
3. import pandas as pd   #data manipulation
4. import matplotlib.pyplot as plt   #visualizing the data
5. import sqlite3   #interacting with the SQL database
6. import seaborn as sns   #complex graphs and visualization
7. #Reading the csv file given and storing it in the pandas database
8. heart\_df = pd.read\_csv('heart.csv')
9. #Split columns will seperate the single column from csv file into different columns where there is a semicolon
10. split\_columns = heart\_df['age;sex;cp;trestbps;chol;fbs;restecg;thalach;exang;oldpeak;slope;ca;thal;target'].str.split(';', expand=True)
11. heart\_df[['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target']] = split\_columns
12. #Removing the column with all the fields together and it won't be represented in the dataframe
13. heart\_df.drop(columns=['age;sex;cp;trestbps;chol;fbs;restecg;thalach;exang;oldpeak;slope;ca;thal;target'], inplace=True)
14. #Connecting to the database created caled heart.db
15. with sqlite3.connect('heart.db') as conn:
16. #Writing data to database and checking table exists
17. heart\_df.to\_sql('heart\_table', conn, if\_exists='replace', index=False)

(Reintech Media, 2023), (Seaborn, 2023), (Markham, 2020)

### Code Output

### Output explanation

The code above performs tasks that are related to data manipulation and database interaction. In the beginning the code imports the necessary libraries and I have explained each reason for the libraries in the comments of my code. The csv file that was provided (“heart.csv”) has been read into a pandas data frame which I have called “heart\_df”. The columns in the data frame are then split based on semicolons and the resulting columns are assigned to the data frame. The original column that contains all the fields together is then dropped. The data frame is written to a SQLite database which I have names “heart.db” which contains a table called “heart\_table”. If the table exists then it will be replaced. The code processes heart-related data from a CSV file, transforms it, as well as stores it in a SQLite database where the data can be analyzed and queried. The output of this code is the creation of a SQLite database file (“heart.db”) which contains a table (“heart\_table”) populated with the processed data from the CSV file.

## Question 2

### 2.1.a.

### Code Snippet

1. #QUESTION 2.1.a
2. #Connecting to the database created in QUESTION 1
3. with sqlite3.connect('heart.db') as conn:
4. #Selecting all the data from the table and reading it
5. query = "SELECT \* FROM heart\_table"
6. heart\_df = pd.read\_sql(query,conn)
7. #Calculating the amount of missing values that each column has and prints it out
8. values\_missing = heart\_df.isnull().sum()
9. print("Missing Values:\n", values\_missing)
10. #Selecting and converting columns to numeric data types
11. num\_columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target']
12. heart\_df[num\_columns] = heart\_df[num\_columns].apply(pd.to\_numeric, errors='coerce')
13. heart\_df.fillna(heart\_df.mean(), inplace=True)

(Omar, 2024), (W3Schools, n.d.)

### Code Output

### Output explanation

The code above performs many tasks related to data handling and cleaning for a dataset stored in a SQLite database named 'heart.db'. A connection to the database is established using the SQLite3 module. A SQL query is executed to select all the data from a table named 'heart\_table' and it is read into a pandas DataFrame named 'heart\_df'. The amount of missing values for each column in the DataFrame is being calculated using the isnull() method and sum() function, and the results are printed out. The specific columns that contain numerical data is selected and they are converted to numeric data types using the pd.to\_numeric() function with the 'coerce' parameter to handle any errors. Missing values are filled in the DataFrame with the mean of each column using the fillna() method with the inplace parameter set to True, which will modify the DataFrame in place. The output of the code gives information on missing values in the dataset and makes sure that the numerical columns are properly converted and any missing values are handled by replacing them with the mean of each column.

### 2.1.b.

### Code Snippet

1. #QUESTION 2.1.b
2. #Defining list using the names of columns
3. categ\_columns = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
4. #Creating a grid
5. fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 10))
6. #Flattening the grid to a 1 Dimension array
7. axes = axes.flatten()
8. #Getting index for each column
9. for i, column in enumerate(categ\_columns):
10. #Creating the plot for the graphs
11. sns.countplot(x=column, hue='target', data=heart\_df, ax=axes[i])
12. #Setting the tittle for each of the graphs
13. axes[i].set\_title(f'Distribution of {column} by Target')
14. #Setting the X label for each of the graphs
15. axes[i].set\_xlabel(column)
16. #Setting the Y label for each of the graphs
17. axes[i].set\_ylabel('Count')
18. #Adjusting the layout of the subplots
19. plt.tight\_layout()
20. plt.show()

(Matplotlib, 2023), (GeeksforGeeks, 2023), (Bobbitt, 2023)

### A group of blue and orange bars Description automatically generatedCode Output

### Output explanation

The matplotlib and seaborn libraries create a grid to visualize the distribution of categorical variables in the dataset ('heart\_df') by the target variable. A list of categorical columns that contains the names of columns of interest is first defined. A grid of subplots with 3 rows and 3 columns is then created which sets the overall size of the figure. A 1-dimensional array is created by flattening the grid for easier iteration. A count plot is created using seaborn's countplot() function, for each column in the list of categorical columns. The x-axis represents the categorical variable, the hue represents the target variable, and the data sourced from 'heart\_df'. Each subplot has set the titles, x labels, and y labels based on the column being visualized. The layout of the subplots is accustomed for better presentation, and the plots are displayed using plt.show(). The code output is a grid of count plots, each of the plots shows the distribution of a categorical variable segmented by the target variable. The visualization will help to understand how different categorical features are distributed with respect to the target variable.

### 2.1.c.

### Code Snippet

1. #QUESTION 2.1.c
2. #Defining the list of names
3. num\_columns = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
4. #Creating and setting up figure
5. fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
6. #Flattening the grid to a 1 Dimension array
7. axes = axes.flatten()
8. #Getting index for each column
9. for i, column in enumerate(num\_columns):
10. #Creating the plot for the graphs
11. sns.boxplot(x='target', y=column, data=heart\_df, ax=axes[i])
12. #Setting the tittle for each of the graphs
13. axes[i].set\_title(f'Distribution of {column} by Target')
14. #Setting the X label for each of the graphs
15. axes[i].set\_xlabel('Target')
16. #Setting the Y label for each of the graphs
17. axes[i].set\_ylabel(column)
18. #Adjusting the layout of the subplots
19. plt.tight\_layout()
20. plt.show()

(Bobbitt, 2023), (GeeksforGeeks, 2023), (Matplotlib, 2023)

### Code Output

### Output explanation

The code for this question will create a grid of subplots using the matplotlib and seaborn that was imported to visualize the distribution of numerical variables in the dataset (‘heart\_df’) by the target variable. A list of numerical columns is first defined, which contains the names of columns of interest. A grid of subplots is then created with 2 rows and 3 columns, which will set up the overall size of the figure. Easier iteration is created when the grid is flattened into a 1-dimensional array. For each column in the list of numerical columns, a box plot will be created using seaborn's boxplot() function (the x-axis represents the target variable, and the y-axis represents the numerical variable) and the data sourced from 'heart\_df'. Titles, X labels, and Y labels will be set for each subplot based on the column being visualized. Better presentation is created by adjusting the layout of the subplots which are also displayed using plt.show(). The output of the code in this question is a grid of box plots, which each shows the distribution of a numerical variable segmented by the target variable. This visualization helps with understanding how different numerical features are distributed with respect to the target variable.

## Question 3

### 3.1.

### Code Snippet

1. #QUESTION 3
2. from sklearn.pipeline import Pipeline
3. from sklearn.compose import ColumnTransformer
4. import joblib
5. from sklearn.ensemble import RandomForestClassifier
6. from sklearn.ensemble import GradientBoostingClassifier
7. from sklearn.impute import SimpleImputer
8. from sklearn.preprocessing import OneHotEncoder
9. from sklearn.preprocessing import StandardScaler
10. from sklearn.metrics import accuracy\_score
11. from sklearn.model\_selection import train\_test\_split
12. from sklearn.svm import SVC
13. #QUESTION 3.1
14. features = heart\_df.drop(columns=['target'])
15. target = heart\_df['target']
16. #Names of categorical features in the dataset in a list
17. categ\_features = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal']
18. #Names of numerical features in the dataset in a list
19. num\_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
20. #Inserting missing values by changing the format and using the value that appears the most in respective columns
21. categ\_transformer = Pipeline(steps=[
22. ('imputer', SimpleImputer(strategy='most\_frequent')),
23. ('onehot', OneHotEncoder())
24. ])
25. #Inserting missing values by standardizing the numerical values that appear the most in respective columns
26. num\_transformer = Pipeline(steps=[
27. ('imputer', SimpleImputer(strategy='mean')),
28. ('scaler', StandardScaler())
29. ])
30. #Preprocessing procedures being integrated for categorical and numerical features
31. preprocessor = ColumnTransformer(
32. transformers=[
33. ('cat', categ\_transformer, categ\_features),
34. ('num', num\_transformer, num\_features)
35. ])
36. #Splitting data into a training set and testing set
37. X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.2, random\_state=42)
38. X\_train\_processed = preprocessor.fit\_transform(X\_train)
39. X\_test\_processed = preprocessor.transform(X\_test)

(Brownlee, 2020), (Pedro, 2022), (Stojiljković, 2021)

### Code Output

### Output explanation

The code above will propare the dataset (‘heart\_df’) for machine learning where several preprocessing steps will be performed and the data will be split into training and testing sets. The target variable ('target') is first separated from the features, and the features are then divided into categorical and numerical lists. For the categorical features, a pipeline will be defined to assign missing values with the most frequent value and apply one-hot encoding. For the numerical features, a pipeline will be defined to assign missing values with the mean and apply standard scaling. A ColumnTransformer combines the transformations which applies the appropriate preprocessing to each feature type. The data is split into training and testing sets, with 80% of the data used for training and 20% for testing. The fit\_transform method is applied to the training features to fit the preprocessing steps and transform the data, and the transform method is applied to the testing features to make sure that the same transformations are applied. The output is two processed datasets (X\_train\_processed and X\_test\_processed) ready for machine learning model training and evaluation, which will make sure that both categorical and numerical features are appropriately preprocessed.

### 3.2.

Three Appropriate machine learning models for the heart disease prediction problem (Le, 2024):

1. Random Forest Classifier - a collaborative learning method which constructs multiple decision trees during training; and produces the mode of the classes (classification) or mean prediction (regression) of the individual trees. It decreases overfitting by averaging multiple decision trees, consequently providing improved generalization.

* Advantages:
  + Random Forests achieve high accuracy and strength by joining the predictions of many decision trees.
  + They offer insights into feature importance, which helps to understand what features contribute most to the prediction.
* Disadvantages:
  + The model can become complex and resource-intensive, which could require considerable memory and computational power.
  + While individual decision trees are easy to interpret, the group of trees in a Random Forest can be difficult to interpret as a whole.

1. Support Vector Machine (SVM) - supervised learning models which investigate data for classification and regression analysis. They find the hyperplane that best divides a dataset into classes. The radial basis function (rbf) kernel maps inputs into higher-dimensional space, which allows the model to handle non-linear relationships.

* Advantages:
  + SVMs are effective in high-dimensional spaces and are memory efficient as a subset of training points is used in the decision function (support vectors).
  + Different kernel functions can be required for the decision function, which provides flexibility in handling various types of data.
* Disadvantages:
  + Training an SVM can be expensive, especially with large datasets involved.
  + The performance of an SVM can be sensitive to the choice of parameters which requires careful tuning.

1. Gradient Boosting Classifier - a collaborative technique which builds models consecutively, where each new model corrects errors made by the last ones. It merges the outputs of many weak learners (typically decision trees) to produce an influential predictive model.

* Advantages:
  + Gradient Boosting often accomplishes high predictive accuracy because of its capability to minimize errors successively.
  + It can be used for classification and regression difficulties and permits for optimization of different loss functions.
* Disadvantages:
  + Gradient Boosting can overfit, especially if the number of trees is too big or if the trees are too deep.
  + The chronological nature of the algorithm can follow in longer training times compared to some other algorithms.

### Code Snippet

1. #QUESTION 3.2
2. #Creating an instance of a RandomForestClassifier from the scikit-learn library
3. random\_forest = RandomForestClassifier(random\_state=42)
4. #Creating an instance of the Support Vector Machine (SVM)
5. svm = SVC(kernel='rbf', random\_state=42)
6. #Creating an instance of the Gradient Boosting Classifier
7. gradient\_boosting\_classifier = GradientBoostingClassifier(random\_state=42)
8. random\_forest.fit(X\_train\_processed, y\_train)  #Training the Random Forest classifier
9. svm.fit(X\_train\_processed, y\_train)  #Training the Support Vector Machine classifier
10. gradient\_boosting\_classifier.fit(X\_train\_processed, y\_train)  #Training the Gradient Boosting classifier
11. rf\_predictions = random\_forest.predict(X\_test\_processed)  #Using the trained Random Forest classifier for predictions
12. svm\_predictions = svm.predict(X\_test\_processed)  #Using the trained Support Vector Machine classifier for predictions
13. gb\_predictions = gradient\_boosting\_classifier.predict(X\_test\_processed)  #Using the trained Gradient Boosting classifier for predictions
14. rf\_accuracy = accuracy\_score(y\_test, rf\_predictions)  #Calculating the accuracy of the Random Forest classifier's predictions
15. svm\_accuracy = accuracy\_score(y\_test, svm\_predictions)  #Calculating the accuracy of the Support Vector Machine classifier's predictions
16. gb\_accuracy = accuracy\_score(y\_test, gb\_predictions)  #Calculating the accuracy of the Gradient Boosting classifier's predictions
17. #Printing the results
18. print(f"Random Forest Accuracy: {rf\_accuracy:.4f}")
19. print(f"SVM Accuracy: {svm\_accuracy:.4f}")
20. print(f"Gradient Boosting Accuracy: {gb\_accuracy:.4f}")
21. #Checking if the Gradient Boosting is most accurate
22. if gb\_accuracy > rf\_accuracy and gb\_accuracy > svm\_accuracy:
23. best\_model = gradient\_boosting\_classifier
24. #If true, the model is saved as a file
25. best\_model\_name = "Gradient\_Boosting\_Model.pkl"
26. joblib.dump(best\_model, best\_model\_name)
27. #Displaying that the best model has been saved to disk
28. print(f"Saved the best model ({best\_model\_name}) to disk.")
29. #Checking if the Random Forrest is most accurate
30. elif rf\_accuracy > svm\_accuracy:
31. best\_model = random\_forest
32. #If true, the model is saved as a file
33. best\_model\_name = "Random\_Forest\_Model.pkl"
34. joblib.dump(best\_model, best\_model\_name)
35. #Displaying that the best model has been saved to disk
36. print(f"Saved the best model ({best\_model\_name}) to disk.")
37. #Checking if the Support Vector Machine is most accurate
38. else:
39. best\_model = svm
40. #If true, then the model is saved as a file
41. best\_model\_name = "SVM\_Model.pkl"
42. joblib.dump(best\_model, best\_model\_name)
43. #Displaying that the best model has been saved to disk
44. print(f"Saved the best model ({best\_model\_name}) to disk.")

(Scikit-learn, 2024), (Tuychiev, 2023), (Snyk Advisor, n.d.)

### Code Output

### Output explanation

The code above executes and assesses three machine learning classifiers (Random Forest, Support Vector Machine (SVM), and Gradient Boosting) for forecasting heart disease. Each of the models is trained on the processed training data and then used to make estimates on the processed test data. The accuracy of each of the model's predictions is calculated and printed. The accuracies of the models are then evaluated to verify which model performs the best. The model with the highest accuracy is saved to disk using joblib. If the Gradient Boosting classifier is the most accurate, it is saved as "Gradient\_Boosting\_Model.pkl". If the Random Forest classifier is the most accurate, it is saved as "Random\_Forest\_Model.pkl". If the SVM is the most accurate, it is saved as "SVM\_Model.pkl". The code outputs the accuracy of each model and confirms which model was saved based on the highest accurateness.

## Question 4

### Code Snippet

1. #QUESTION 4
2. #Importing necessary libraries
3. import streamlit as st  #Streamlit for creating the web app
4. import pandas as pd  #pandas for data manipulation
5. import joblib  #joblib for loading the trained model
6. from sklearn.preprocessing import StandardScaler  #StandardScaler for feature scaling
7. #Loading the trained SVM model from a file
8. model = joblib.load('SVM\_Model.pkl')
9. #Creating an instance of StandardScaler for scaling the input data
10. data\_scaler = StandardScaler()
11. #Setting the title of the Streamlit app
12. st.title('Heart Disease Prediction')
13. #Writing a description of the app
14. st.write('Enter patient details to predict if they have heart disease.')
15. #Function to preprocess the input data
16. def preprocess\_input\_data(patient\_age, patient\_sex, chest\_pain, resting\_bp, cholesterol, fasting\_bs, ecg\_result, max\_heart\_rate, exercise\_angina, st\_depression, st\_slope, major\_vessels, thalassemia):
17. patient\_sex = 1 if patient\_sex == 'Male' else 0  #Converting sex to binary value (1 for Male, 0 for Female)
18. fasting\_bs = 1 if fasting\_bs == 'True' else 0  #Converting fasting blood sugar to binary value (1 for True, 0 for False)
19. #Creating a DataFrame with the input data
20. input\_data = pd.DataFrame({
21. 'age': [patient\_age],
22. 'sex': [patient\_sex],
23. 'cp': [chest\_pain],
24. 'trestbps': [resting\_bp],
25. 'chol': [cholesterol],
26. 'fbs': [fasting\_bs],
27. 'restecg': [ecg\_result],
28. 'thalach': [max\_heart\_rate],
29. 'exang': [exercise\_angina],
30. 'oldpeak': [st\_depression],
31. 'slope': [st\_slope],
32. 'ca': [major\_vessels],
33. 'thal': [thalassemia]
34. })
35. return input\_data
36. #Input fields for the user to enter patient details
37. patient\_age = st.number\_input('Age', min\_value=1, max\_value=120)  #Age input field
38. patient\_sex = st.radio('Sex', ['Male', 'Female'])  #Sex input field
39. chest\_pain = st.selectbox('Chest Pain Type', ['Typical Angina', 'Atypical Angina', 'Non-anginal Pain', 'Asymptomatic'])  #Chest pain type input field
40. resting\_bp = st.number\_input('Resting Blood Pressure (mmHg)', min\_value=1)  #Resting blood pressure input field
41. cholesterol = st.number\_input('Cholesterol (mg/dl)', min\_value=1)  #Cholesterol input field
42. fasting\_bs = st.radio('Fasting Blood Sugar > 120 mg/dl', ['True', 'False'])  #Fasting blood sugar input field
43. ecg\_result = st.selectbox('Resting ECG Result', ['Normal', 'ST-T Wave Abnormality', 'Left Ventricular Hypertrophy'])  #Resting ECG result input field
44. max\_heart\_rate = st.number\_input('Max Heart Rate Achieved', min\_value=1)  #Max heart rate achieved input field
45. exercise\_angina = st.radio('Exercise Induced Angina', ['Yes', 'No'])  #Exercise induced angina input field
46. st\_depression = st.number\_input('ST Depression Induced by Exercise Relative to Rest', min\_value=0.0)  #ST depression input field
47. st\_slope = st.selectbox('Slope of the Peak Exercise ST Segment', ['Upsloping', 'Flat', 'Downsloping'])  #Slope of the peak exercise ST segment input field
48. major\_vessels = st.selectbox('Number of Major Vessels Colored by Flourosopy', [0, 1, 2, 3])  #Number of major vessels input field
49. thalassemia = st.selectbox('Thalassemia', ['Normal', 'Fixed Defect', 'Reversible Defect'])  #Thalassemia input field
50. #Predict button
51. if st.button('Predict'):
52. #Preprocessing the input data
53. input\_data = preprocess\_input\_data(patient\_age, patient\_sex, chest\_pain, resting\_bp, cholesterol, fasting\_bs, ecg\_result, max\_heart\_rate, exercise\_angina, st\_depression, st\_slope, major\_vessels, thalassemia)
54. #Scaling the input data
55. input\_data\_scaled = data\_scaler.fit\_transform(input\_data)
56. #Making predictions using the loaded model
57. prediction = model.predict(input\_data\_scaled)
58. #Displaying the prediction result
59. if prediction[0] == 1:
60. st.success('The patient is likely to have heart disease.')
61. else:
62. st.success('The patient is unlikely to have heart disease.')

(GeeksforGeeks, 2023)

### A screenshot of a computer Description automatically generatedA screenshot of a computer Description automatically generatedCode Output

### Output explanation

The code executes a web application using Streamlit where heart disease will be predicted based on what the user inputs. When run, the application asks the user to enter various health-related metrics such as age, gender, chest pain type, resting blood pressure, cholesterol level, and several other factors associated with heart disease. These inputs are pre-processed and scaled before being fed into a pre-trained Support Vector Machine (SVM) model loaded from a file named SVM\_Model.pkl. This was chosen in the previous question 3.2. When the "Predict" button is clicked, the app will make a prediction about whether the user is likely to have heart disease. If the model predicts a positive outcome (i.e., the patient is likely to have heart disease), the app will display a success message indicating this likelihood. Otherwise, it displays a success message indicating that the patient is unlikely to have heart disease. This provides a simple yet interactive way for the users to receive a prediction based on their health parameters.

# References

Bobbitt, Z., 2023. *How to Use tight\_layout() in Matplotlib.* [Online]   
Available at: https://www.statology.org/matplotilb-tight\_layout/  
[Accessed 15 May 2024].

Brownlee, J., 2020. *How to Use the ColumnTransformer for Data Preparation.* [Online]   
Available at: https://machinelearningmastery.com/columntransformer-for-numerical-and-categorical-data/  
[Accessed 16 May 2024].

GeeksforGeeks, 2023. *A Beginners Guide To Streamlit.* [Online]   
Available at: https://www.geeksforgeeks.org/a-beginners-guide-to-streamlit/  
[Accessed 20 May 2024].

GeeksforGeeks, 2023. *Numpy ndarray.flatten() function | Python.* [Online]   
Available at: https://www.geeksforgeeks.org/numpy-ndarray-flatten-function-python/  
[Accessed 15 May 2024].

Le, J., 2024. *The Top 10 Machine Learning Algorithms to Know.* [Online]   
Available at: https://builtin.com/data-science/tour-top-10-algorithms-machine-learning-newbies  
[Accessed 17 May 2024].

Markham, K., 2020. *How to Drop Columns in Pandas Tutorial.* [Online]   
Available at: https://www.datacamp.com/tutorial/pandas-drop-column  
[Accessed 10 May 2024].

Matplotlib, 2023. *matplotlib.pyplot.subplots.* [Online]   
Available at: https://matplotlib.org/stable/api/\_as\_gen/matplotlib.pyplot.subplots.html  
[Accessed 15 May 2024].

Omar, 2024. *Identify missing values in each column with pandas..* [Online]   
Available at: https://medium.com/@bouimouass.o/identify-missing-values-in-each-column-with-pandas-d4bdefb150dd  
[Accessed 12 May 2024].

Pedro, J., 2022. *Improve Your Data Preprocessing with ColumnTransformer and Pipelines.* [Online]   
Available at: https://towardsdatascience.com/improve-your-data-preprocessing-with-columntransformer-and-pipelines-b6ff7edd2f77  
[Accessed 16 May 2024].

Reintech Media, 2023. *How to Connect to a Database with Python.* [Online]   
Available at: https://reintech.io/blog/connect-to-database-with-python  
[Accessed 10 May 2024].

Scikit-learn, 2024. *SVC.* [Online]   
Available at: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html  
[Accessed 18 May 2024].

Seaborn, 2023. *An introduction to seaborn.* [Online]   
Available at: https://seaborn.pydata.org/tutorial/introduction.html  
[Accessed 9 May 2024].

Snyk Advisor, n.d. *How to use the sklearn.ensemble.RandomForestClassifier function in sklearn.* [Online]   
Available at: https://snyk.io/advisor/python/sklearn/functions/sklearn.ensemble.RandomForestClassifier  
[Accessed 19 May 2024].

Stojiljković, M., 2021. *Split Your Dataset With scikit-learn's train\_test\_split().* [Online]   
Available at: https://realpython.com/train-test-split-python-data/  
[Accessed 16 May 2024].

Tuychiev, B., 2023. *A Guide to The Gradient Boosting Algorithm.* [Online]   
Available at: https://www.datacamp.com/tutorial/guide-to-the-gradient-boosting-algorithm  
[Accessed 18 May 2024].

W3Schools, n.d. *Pandas DataFrame fillna() Method.* [Online]   
Available at: https://www.w3schools.com/python/pandas/ref\_df\_fillna.asp  
[Accessed 13 May 2024].